Classification of Fashion Article Images using Convolutional Neural Networks

Shobhit Bhatnagar*
Indian Institute of Technology Patna
shobhit.bhat@gmail.com

Deepanway Ghosal*
Indian Institute of Technology Patna
deepanwayedu@gmail.com

Maheshkumar H. Kolekar
Indian Institute of Technology Patna
mkolekar@gmail.com

***Abstract*—In this paper, we propose a state-of-the-art model for classification of fashion article images. We trained convolutional neural network based deep learning architectures to classify images in the Fashion-MNIST dataset. We have proposed three different convolutional neural network architectures and used batch normalization and residual skip connections for ease and acceleration of learning process. Our model shows impressive results on the benchmark dataset of Fashion-MNIST. Comparisons show that our proposed model reports improved accuracy of around 2% over the existing state-of-the-art systems in literature.**

***Index Terms*—Deep Learning, Object Classification, Convolutional Neural Network (CNN), Fashion MNIST.**

I. Introduction

This paper demonstrates the use of Convolutional Neural Networks for image classification of the Fashion-MNIST dataset. Fashion-MNIST is a dataset of Zalando's fashion article images having a training set of 60,000 examples and a test set of 10,000 examples [1]. Each example is a 28x28 grayscale image. Each image is associated with a label from 10 classes as shown in the figure 1.

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig. 1: Fashion-MNIST Dataset

* The first two authors have contributed equally to this work.

II. Problem Definition

Image classification is one of the most foundational problems in computer vision, which has a variety of practical applications such as image and video indexing [2] [3]. Although the problem of identifying a visual entity from an image is a very trivial problem for a human-being to perform, it is very challenging for a computer algorithm to do the same with human level accuracy [4] [5]. The algorithm must be invariant to a number of variations in order to successfully identify and classify the images. For example, different illumination conditions, different scale and viewpoint variations, deformations, occlusions may influence the algorithm to wrongly predict the image class.

In recent times, deep neural networks have been applied to a multitude of problems to achieve very good performances. In particular, convolutional neural networks have shown very good results in image classification [6], image segmentation [7], computer vision problems [8] [9] and natural language processing problems [10]. Some probabilistic models based on Bayesian Belief Networks [11] and Hidden Markov Models [12] [13] have also been applied to image classification problems with features based on grey level, color, motion, depth, and texture [14]. In this paper we explore the idea of classifying Fashion MNIST images with variants of convolutional neural networks.

III. Proposed Methodology

Convolutional neural networks are neuro-biologically inspired. A typical layer of a convolutional network consists of three stages. In the first stage we use a number (tens to thousands) of filters or kernels of normally very small dimension, generally 3x3, 4x4 or 5x5 and slide it over the input image to create a feature map [15]. As we slide the kernel over the image we add up the element wise dot product of the filter values and the section of the image it is sliding over. As the same kernel is operated over the image, it is a very memory efficient operation. As the kernels used in a layer are independent of each other, results can be computed extremely fast in a graphical processing unit (GPU). The convolution operation between a two dimensional image I and a two dimensional kernel K is,

$$S(i,j) = (I*K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \quad (1)$$

978-1-5090-6734-3/17/$31.00 ©2017 IEEE

It can be rewritten as,

$$S(i,j) = (K*I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \quad (2)$$

For a 3*3 kernel size the equation becomes,

$$S(i,j) = (K*I)(i,j) = \sum_{m=1}^{3} \sum_{n=1}^{3} I(i-m, j-n)K(m,n) \quad (3)$$

In next stage, we apply a non-linear activation function over the resultant dot product. Usage of 'tanh' and 'ReLU' activation is mostly found in literature [16]. The 'tanh' operation is,

$$A(i,j) = tanh(S(i,j)) = \frac{sinh(S(i,j))}{cosh(S(i,j))} = \frac{1 - e^{-2*S(i,j)}}{1 + e^{-2*S(i,j)}} \quad (4)$$

and the 'ReLU' operation is,

$$A(i,j) = \begin{cases} 0 & \text{if } S(i,j) < 0 \\ S(i,j) & \text{if } S(i,j) \geq 0 \end{cases} \quad (5)$$

Finally, a pooling function is applied to the 'tanh' or 'ReLU' activated output of the net at every sliding location with a summary statistic of the nearby outputs. Generally, max-pooling or average-pooling is used [15], which down samples the image into a smaller dimension. To demonstrate, if

$$A(i,j) = \begin{bmatrix} 10 & 20 & 120 & 0 \\ 30 & 40 & 0 & 100 \\ 255 & 255 & 90 & 10 \\ 255 & 255 & 200 & 0 \end{bmatrix}$$

then a 2x2 max-pooling and average pooling operation would result in,

$$MaxP(i,j) = \begin{bmatrix} 40 & 120 \\ 255 & 200 \end{bmatrix} \quad AvgP(i,j) = \begin{bmatrix} 25 & 55 \\ 255 & 75 \end{bmatrix}$$

If the network is several layers deep the the next convolution operation will be done on the max or average pooled output. After performing the series of convolution, activation and pooling operations the output is fed into a multilayer perceptron network to classify the image. This sliding kernel convolution in the first stage and the pooling operation in the third stage are locally invariant and thus are very useful for detecting and classifying objects present in the image. The non-linearity in the second stage enables the network to become more powerful universal function approximators.

A. Convolutional Neural Network

In this model, we used 2 convolutional and max pooling layers one after the other. Each convolutional layer has 32 filters of size 3x3, and max pooling were performed on every 2x2 pixels. The output of this was flattened and fed into a multi layer perceptron network for classification.

B. Using Batch Normalization

In this model, Batch Normalization [17] was done before every convolutional layer to improve the training speed of the model. The key idea is to normalize the inputs to layers within the network along with the input to the whole network. During the training process, we normalize each convolution layers inputs by using the mean and variance of the values in the currently processing mini batch. The mini batch input is generally modified to have zero mean and unit variance. This technique enables us to train the network faster with higher learning rates. With Batch Normalization we were able to achieve the same loss function value after 10 epochs which was achieved after 40 epochs without Batch Normalization.

C. Using Residual Skip Connections

When a deep neural network is being trained via backpropagation, the gradient signal must be propagated backwards from the topmost output layer all the way down to the bottommost input layer to ensure that the parameters are updated appropriately. Residual skip connections are an intuitive way to achieve this objective [18]. In this model we add the previous input and current value of convoluted output to get the final output. Precisely for a 3x3 kernel it can be written as,

$$y = convolution(x) + x \quad (6)$$

$$S(i,j) = \sum_{m=1}^{3} \sum_{n=1}^{3} I(i-m, j-n)K(m,n) + I(i,j) \quad (7)$$

D. Other Network Parameters

After flattening the output of the last max-pooling layer in each of the models we use a multi layer perceptron (MLP) network to classify the images. We use a hidden layer having 128 neurons. The final output layer has 10 neurons for the 10 different classes. We used 'ReLU' activation function in the hidden layer and a softmax activation function in the final layer. The softmax function, or normalized exponential function, transforms a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range [0, 1] that add up to 1. The function is given by,

$$\phi(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad for \quad j = 1, .., K \quad (8)$$

As, after the transformation the values add to 1, we can use the values as the predicted probability distribution for a K class classification problem.

We also use 50% dropout [19] as a measure of regularization. As our convolution nets have a fairly large number of trainable parameters, it is quite possible the model highly overfits to the training data and generalizes poorly for the test data. Dropout is a technique which addresses this overfitting problem by randomly dropping out neurons during the training process. This prevents the network from over-relying on certain neurons which increase the generalization capability of the

model. We have used the categorical cross entropy function as our error estimate or loss function. Categorical cross entropy is defined as,

$$H_p(y) = -\sum_i y_i * \log p_i \tag{9}$$

where p_i is the predicted probability distribution of class i and y_i is the true probability distribution of class i. As we have ten classes in our dataset the summation is computed over 10 terms. After every minibatch iteration of the training process we compute the total cross entropy error and optimize the parameters of the network so that the loss is gradually minimized. We have used the 'Adam' optimizer [20] for optimization of the loss function. Fig. 2 and 3 shows how the training loss is decreased and testing accuracy in increased with increasing epochs, respectively. An architectural summary of all the models has been shown in Fig 4.

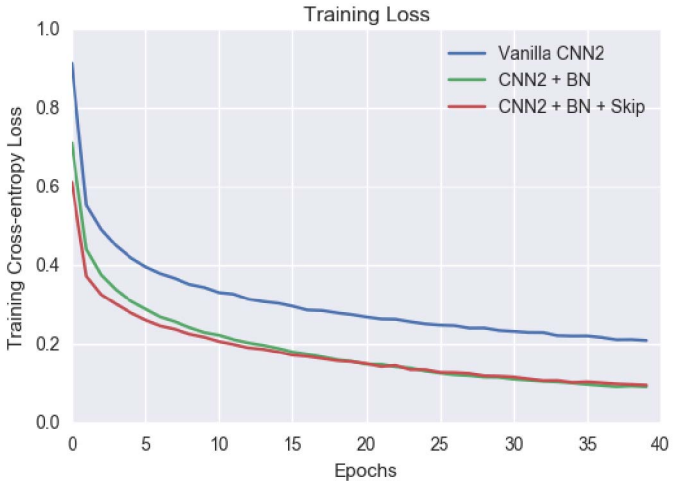

Fig. 2: Training Loss vs Epoch

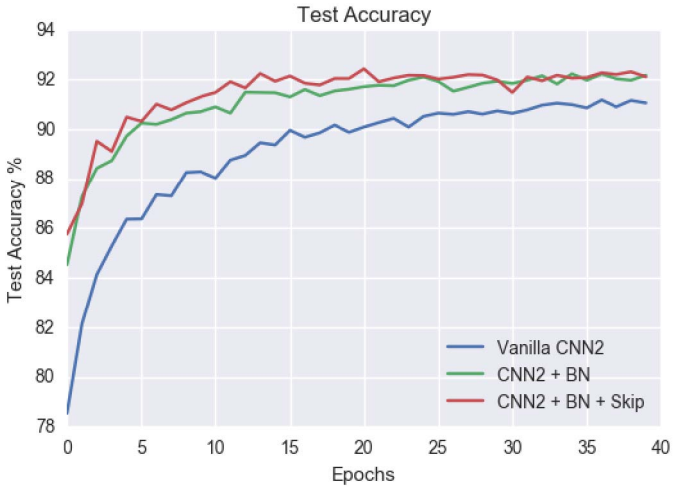

Fig. 3: Test Accuracy vs Epoch

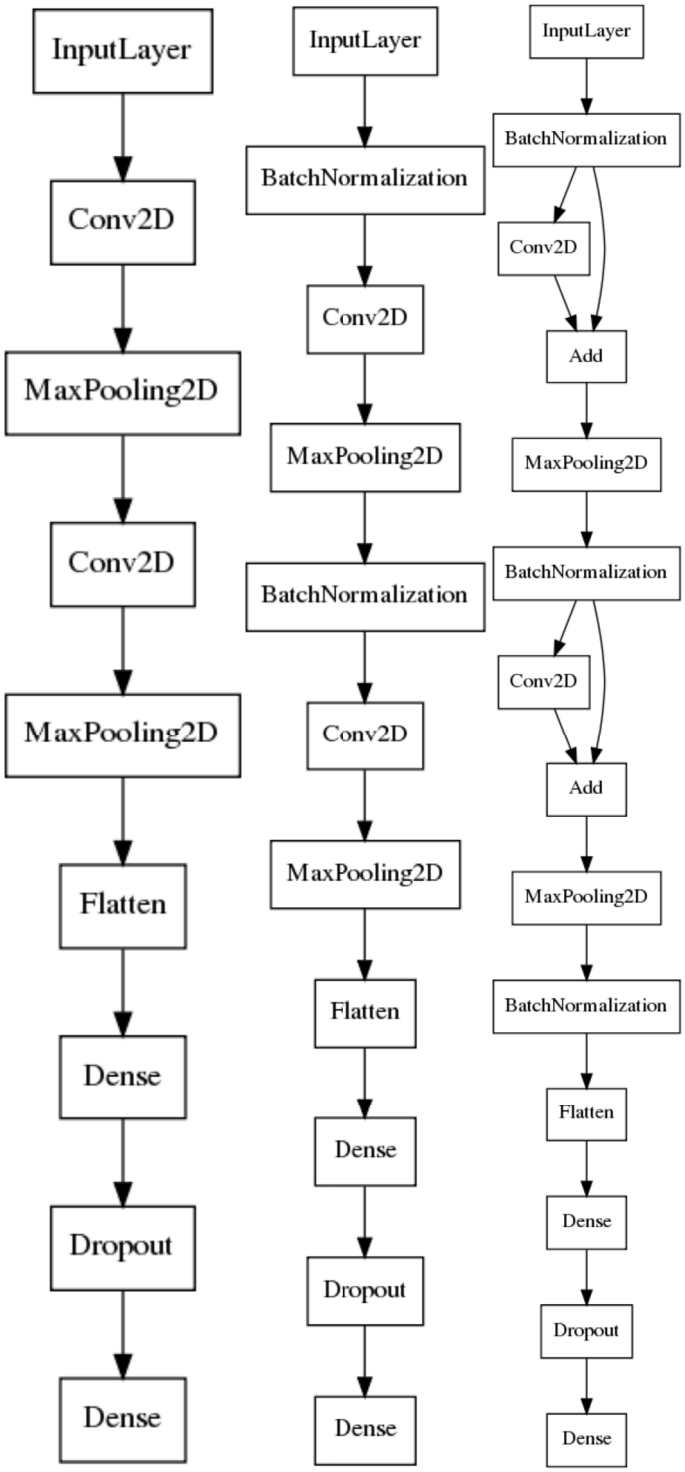

Fig. 4: Model Architectures

IV. Results

We compare our results with best reported models [1], [21] in literature. In [1], support vector classifier (SVC) with rbf kernel is used whereas in [21] an evolutionary deep learning framework is used. The results of [1] and [21] are shown in

Table I. Our proposed CNN based classifiers' performance are summarized in Table II.

Models	Test Accuracy (%)
SVC C = 10; kernel : rbf [1]	89.70
EDEN [21]	90.60

TABLE I: Accuracy on Fashion MNIST test data for existing state of the art models in literature. SVC: Support Vector Classifier; EDEN: Evolutionary Deep Learning.

Models	Test Accuracy (%)
CNN2	91.17
CNN2 + BatchNorm	92.22
CNN2 + BatchNorm + Skip	92.54

TABLE II: Accuracy on Fashion MNIST test data for our models. CNN2: 2 convolutional and max-pool layers; BatchNorm: BatchNormalization; Skip: Residual Skip Connections.

V. Visualization

Various approaches for visualization and understanding of convolutional nets has been developed in literature [22]. One of the most popular strategy is to visualize the convolution layer outputs of an input image during forward propagation. This method will generally show different boundaries, corners in lower level convolutional layers, and gradually more higher level features in the deeper convolutional layers. Another very common strategy is to visualize the weights of the network. This is useful because well-trained networks usually display smooth and nice kernels without any noisy patterns. Noisy structures can indicate that the network most probably has not been trained for enough time, or possibly a low regularization factor which have led to overfitting.

In this work, we follow the first approach i.e. we visualize the convolution layer outputs for a few input images. We take the CNN2 + BatchNorm + Skip model which has 2 convolution layers. We pass an image from the 'Pullover' class and another image from the 'Bag' class to the network as shown in Fig. 5. Both the convolution layers have 32 different kernels, so we have 32 different channels for each image in each convolution layer. The outputs of the two convolution layers for the two images from Pullover and Bag class are shown in Fig. 6 and Fig. 7.

Some very interesting results can be observed in the convolution layer outputs for the images. For example, in the eighth channel in the first layer of Fig. 6, we observed that, the pixels are high for the sleeves part of the pullover, rest of the pixels are relatively low, so this is effectively working as a 'sleeve detector'. For the image of the bag, we observed that in the first channel in the first layer in Fig. 7, the pixels are high for the strap of the bag, but the rest of the pixels are low. So, this is essentially working as a 'strap detector'. We observed similar patterns in the rest of the image classes too. Another interesting thing which can be observed is that, for any input image, outputs of the lower layer convolutional layers are more sharper in nature, and becomes gradually blurred in the higher level convolutional layers. This is analogous to our understanding of deep convolutional nets, where lower levels of the network see small portion of the image and as we do max-pooling, consequent higher levels of the network is exposed to more and more parts of the image. This is how deeper convolutional layers is able to use the learned higher level features to classify the images.

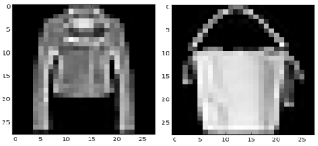

Fig. 5: Two input images from the 'Pullover' and 'Bag' class.

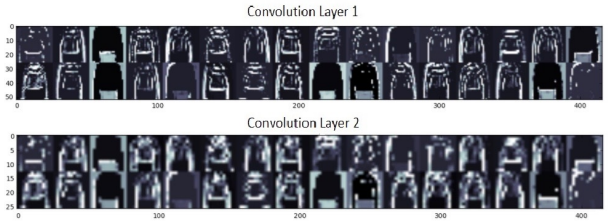

Fig. 6: Convolution layer outputs for the pullover image.

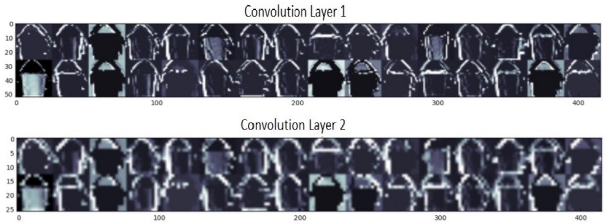

Fig. 7: Convolution layer outputs for the bag image.

VI. Error Analysis

For all the analysis performed and described in this section, we took the predictions from the CNN2 + BatchNorm + Skip model, as this gave the best performance on the test data. We perform both qualitative and quantitative error analysis on the obtained results and observe a few interesting error patterns. For instance, the six most wrongly classified images i.e. the examples with highest categorical cross entropy error between the actual classes and the predicted probability distribution are shown in Fig. 8 and Table. III. We observe that the images 8(a), 8(b), 8(f) actually belongs to the Sneaker, Ankle boots and Sneaker class, but are misclassified as Ankle boots, Sandals and Ankle boots class. Misclassification between these three particular class is a fairly common trend among the wrongly

predicted images. Similarly the image 8(d) can seem like a gown or dress to the computer, whereas, it actually is a trouser. Images in Shirt, Coat, T-Shirt/Top, Pullover class are also mistaken interchangeably. A couple of good examples of this are the 8(c) and 8(e) images.

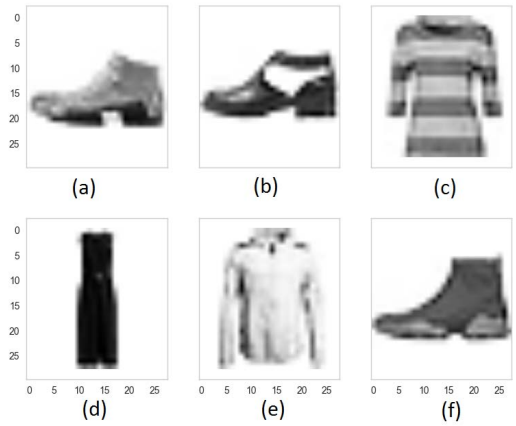

Fig. 8: Images predicted with most cross-entropy error.

Image	Predicted Class	Actual Class
Fig. 8 (a)	Ankle Boots	Sneaker
Fig. 8 (b)	Sandals	Ankle Boots
Fig. 8 (c)	T-Shirt/Top	Pullover
Fig. 8 (d)	Dress	Trouser
Fig. 8 (e)	Shirt	Coat
Fig. 8 (f)	Ankle Boots	Sneaker

TABLE III: Images predicted with most cross-entropy error.

This error pattern becomes more evident from the classification report shown in Table. IV. Precision, Recall are F1-Score are widely used metrics for classification problems. These are defined as:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

We can see that, the all the scoring metrics viz. precision, recall and f1 score are somewhat low for the T-shirt/Top, Pullover, Coat, Shirt classes compared to the rest of the classes. Images of these four classes are most often misclassified among the other three classes. One possible reason may could be that, 28x28 grayscale images are not really sufficient for the algorithm to differentiate between these four classes with very good accuracy, because these classes will look mostly similar in such small images. Distinguishing features like colour, texture are only available in rgb images, and the algorithm will probably be able to leverage those informations and predict correctly between these classes only in coloured images.

Class	Class Label	Precision	Recall	F1 Score
T-Shirt/Top	0	0.88	0.86	0.87
Trouser	1	0.99	0.98	0.99
Pullover	2	0.90	0.85	0.88
Dress	3	0.92	0.93	0.92
Coat	4	0.87	0.88	0.88
Sandals	5	0.99	0.99	0.99
Shirt	6	0.75	0.80	0.77
Sneaker	7	0.96	0.97	0.97
Bag	8	0.99	0.99	0.99
Ankle Boots	9	0.98	0.97	0.97
Overall	-	0.92	0.92	0.92

TABLE IV: Precision, Recall and F1 Score for all classes. Class labels 0, 1,.., 9 are used in the confusion matrix plot in Fig 9.

We conclude this section by plotting the confusion matrix in Fig 9. From the plot it is clear that misclassified images from the class 0 (T-Shirt/Top), 2 (Pullover), 4 (Coat) and 6 (Shirt) are mostly the sources of error. 100 images actually belonging to the T-Shirt/Top class are wrongly predicted as Coat class. Similarly 83 images belonging to the Coat class are wrongly predicted as T-Shirt/Top class. This two misclassifications are the two biggest source of error in our model.

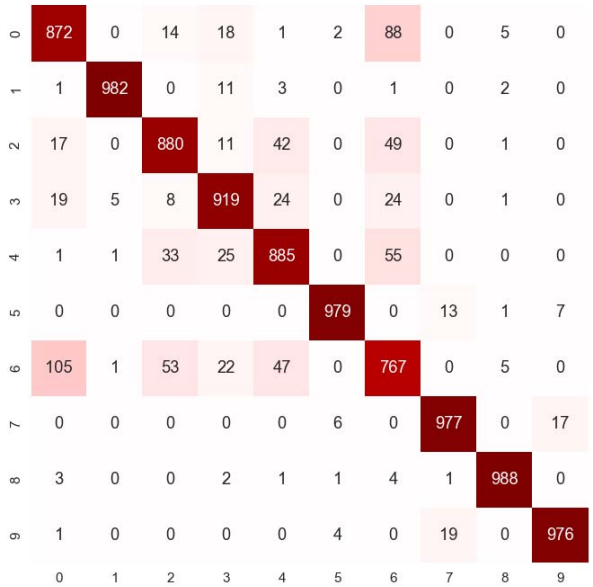

Fig. 9: Confusion Matrix: CNN2 + BatchNorm + Skip model.

VII. Conclusion

We were able to achieve an accuracy of 92.54% by using a two layer CNN along with Batch Normalization and Skip Connections. We can clearly see how Batch Normalization and Skip Connections help improve the overall accuracy and significantly reduce the training time. Identifying an article type in the fashion industry is a crucial task. This model can serve as a building block for a service which shows related

fashion articles based on a given image. This work can also be extended to image and video indexing [23].

REFERENCES

[1] Xiao, H., Rasul, K. and Vollgraf, R., 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

[2] Kolekar, M.H., Palaniappan, K., Sengupta, S. and Seetharaman, G., 2009. Semantic Concept Mining Based on Hierarchical Event Detection for Soccer Video Indexing. Journal of multimedia, 4(5).

[3] Kolekar, M.H., 2011. Bayesian belief network based broadcast sports video indexing. Multimedia Tools and Applications, 54(1), pp.27-54.

[4] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[5] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE.

[6] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[7] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[8] Gatys, L.A., Ecker, A.S. and Bethge, M., 2015. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

[9] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A.A., 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In AAAI (pp. 4278-4284).

[10] Ghosal, D., Bhatnagar, S., Akhtar, M.S., Ekbal, A. and Bhattacharyya, P., 2017. IITP at SemEval-2017 task 5: an ensemble of deep learning and feature based models for financial sentiment analysis. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 899-903).

[11] Kolekar, M.H. and Sengupta, S., 2015. Bayesian network-based customized highlight generation for broadcast soccer videos. IEEE Transactions on Broadcasting, 61(2), pp.195-209.

[12] Kolekar, M.H. and Sengupta, S., 2004, December. Hidden markov model based video indexing with discrete cosine transform as a likelihood function. In India Annual Conference, 2004. Proceedings of the IEEE INDICON 2004. First (pp. 157-159). IEEE.

[13] Kolekar, M.H. and Sengupta, S., 2004. Hidden Markov Model Based Structuring of Cricket Video Sequences Using Motion and Color Features. In ICVGIP (pp. 632-637).

[14] Kolekar, M.H., Talbar, S.N. and Sontakke, T.R., 2000. Texture segmentation using fractal signature. IETE Journal of Research, 46(5), pp.319-323.

[15] Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.

[16] Nair, V. and Hinton, G.E., 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10) (pp. 807-814).

[17] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning. 2015.

[18] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[19] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research, 15(1), pp.1929-1958.

[20] Kingma, D. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[21] Dufourq, E. and Bassett, B.A., 2017. EDEN: Evolutionary Deep Networks for Efficient Machine Learning. arXiv preprint arXiv:1709.09161.

[22] Zeiler, M.D. and Fergus, R., 2014, September. Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[23] Kolekar, M.H. and Sengupta, S., 2005, November. Semantic indexing of news video sequences: a multimodal hierarchical approach based on hidden markov model. In TENCON 2005 2005 IEEE Region 10 (pp. 1-6). IEEE.